

## APPENDIX A

### EXAMPLE COMMAND ALIAS

In an exemplary system environment **200** as shown in **Fig. 2**, the following Command Alias file is located in the #pragma Namespace ("\\.\root\ops") on a management station **202**:

**Example:** (Win32\_NetworkAdapter class command alias)

```
instance of Microsoft_CliAlias
{
    Connection =
    instance of Microsoft_CliConnection
    {
        Locale = "ms_409";
        NameSpace = "ROOT\\CIMV2";
        Server = "BAMBAZONKI";
    };
    Descriptions = {
        instance of Microsoft_CliLocalizedString
        {
            CodePage = 401;
            Text = "The Win32_NetworkAdapter class represents a
network adapter on a Win32 system.";
        }};
    Formats = {
        instance of Microsoft_CliFormat
        {
            Name = "FULL";
            Properties = {
                instance of Microsoft_CliProperty
                {
                    Derivation = "AdapterType";
                    Descriptions = {
                        instance of
Microsoft_CliLocalizedString
                        {
                            CodePage = 401;
                            Text = "The AdapterType
property reflects the network medium in use. This property may not be
applicable to all types of network adapters listed within this class.
Windows NT only.";
                        }};
                    Name = "AdapterType";
                },
                instance of Microsoft_CliProperty
                {
```

```

Derivation = "AutoSense";
Descriptions = {
    instance of
5  Microsoft_CliLocalizedString
    {
        CodePage = 401;
        Text = "A boolean indicating
whether the network adapter is capable of automatically determining the
speed of the attached/network media.";
10    }
    };
    Name = "AutoSense";
    },
    instance of Microsoft_CliProperty
    {
15        Derivation = "Availability";
        Descriptions = {
            instance of
Microsoft_CliLocalizedString
20            {
                CodePage = 401;
                Text = "The availability and
status of the device. For example, the Availability property indicates
that the device is running and has full power (value=3), or is in a
warning (4), test (5), degraded (10) or power save state (values 13-15
and 17). Regarding the power saving states, these are defined as
25 follows: Value 13 (\\"Power Save - Unknown\\") indicates that the device
is known to be in a power save mode, but its exact status in this mode
is unknown; 14 (\\"Power Save - Low Power Mode\\") indicates that the
device is in a power save state but still functioning, and may exhibit
30 degraded performance; 15 (\\"Power Save - Standby\\") describes that the
device is not functioning but could be brought to full power 'quickly';
and value 17 (\\"Power Save - Warning\\") indicates that the device is in
a warning state, though also in a power save mode.";
35    }
    };
    Name = "Availability";
    },
    instance of Microsoft_CliProperty
    {
40        Derivation = "Caption";
        Descriptions = {
            instance of
Microsoft_CliLocalizedString
45            {
                CodePage = 401;
                Text = "The Caption property
is a short textual description (one-line string) of the object.";
            }
    };
    Name = "Caption";
    },
50    instance of Microsoft_CliProperty
    {
        Derivation = "ConfigManagerErrorCode";
        Descriptions = {
            instance of
55 Microsoft_CliLocalizedString
            {

```

```

CodePage = 401;
Text = "Indicates the Win32
Configuration Manager error code.";
    });
5     Name = "ConfigManagerErrorCode";
    },
    instance of Microsoft_CliProperty
    {
10     Derivation = "ConfigManagerUserConfig";
    Descriptions = {
        instance of
Microsoft_CliLocalizedString
        {
15     CodePage = 401;
    Text = "Indicates whether the
device is using a user-defined configuration.";
    });
    Name = "ConfigManagerUserConfig";
    },
20     instance of Microsoft_CliProperty
    {
    Derivation = "CreationClassName";
    Name = "CreationClassName";
    },
25     instance of Microsoft_CliProperty
    {
    Derivation = "Description";
    Descriptions = {
        instance of
30     Microsoft_CliLocalizedString
        {
    CodePage = 401;
    Text = "The Description
property provides a textual description of the object. ";
35     });
    Name = "Description";
    },
    instance of Microsoft_CliProperty
    {
40     Derivation = "DeviceID";
    Descriptions = {
        instance of
Microsoft_CliLocalizedString
        {
45     CodePage = 401;
    Text = "The DeviceID property
contains a string uniquely identifying the network adapter from other
devices on the system.";
    });
50     Name = "DeviceID";
    },
    instance of Microsoft_CliProperty
    {
55     Derivation = "ErrorCleared";
    Descriptions = {

```



```

        Descriptions = {
            instance of
Microsoft_CliLocalizedString
5
            {
                CodePage = 401;
                Text = "The Installed property
determines whether the network adapter is installed in the
system.\nValues: TRUE or FALSE. A value of TRUE indicates the network
10 adapter is installed.";
            }
            Name = "Installed";
        },
        instance of Microsoft_CliProperty
15
        {
            Derivation = "LastErrorCode";
            Descriptions = {
                instance of
Microsoft_CliLocalizedString
20
                {
                    CodePage = 401;
                    Text = "LastErrorCode captures
the last error code reported by the logical device.";
                }
            }
            Name = "LastErrorCode";
25
        },
        instance of Microsoft_CliProperty
        {
            Derivation = "MACAddress";
            Descriptions = {
30
                instance of
Microsoft_CliLocalizedString
        {
            CodePage = 401;
            Text = "The MACAddress
35 property indicates the media access control address for this network
adapter. A MAC address is a unique 48-bit number assigned to the network
adapter by the manufacturer. It uniquely identifies this network adapter
and is used for mapping TCP/IP network communications.";
        }
            Name = "MACAddress";
40
        },
        instance of Microsoft_CliProperty
        {
            Derivation = "Manufacturer";
            Descriptions = {
45
                instance of
Microsoft_CliLocalizedString
        {
            CodePage = 401;
            Text = "The Manufacturer
50 property indicates the name of the network adapter's
manufacturer.\nExample: 3COM.";
        }
            Name = "Manufacturer";
55
        },
        instance of Microsoft_CliProperty

```

```

{
    Derivation = "MaxNumberControlled";
    Descriptions = {
        instance of
5   Microsoft_CliLocalizedString
        {
            CodePage = 401;
            Text = "The
10  MaxNumberControlled property indicates the maximum number of directly
    addressable ports supported by this network adapter. A value of zero
    should be used if the number is unknown.";
            }
        };
        Name = "MaxNumberControlled";
    },
15  instance of Microsoft_CliProperty
    {
        Derivation = "MaxSpeed";
        Descriptions = {
            instance of
20  Microsoft_CliLocalizedString
                {
                    CodePage = 401;
                    Text = "The maximum speed, in
25  bits per second, for the network adapter.";
                }
            };
            Name = "MaxSpeed";
        },
        instance of Microsoft_CliProperty
        {
30  Derivation = "Name";
        Descriptions = {
            instance of
            Microsoft_CliLocalizedString
                {
35  CodePage = 401;
                    Text = "The Name property
                    defines the label by which the object is known. When subclassed, the
                    Name property can be overridden to be a Key property.";
                }
            };
            Name = "Name";
        },
        instance of Microsoft_CliProperty
        {
45  Derivation = "NetworkAddresses";
        Descriptions = {
            instance of
            Microsoft_CliLocalizedString
                {
50  CodePage = 401;
                    Text = "An array of strings
                    indicating the network addresses for an adapter.";
                }
            };
            Name = "NetworkAddresses";
        },
55  instance of Microsoft_CliProperty
        {

```

```

Derivation = "PermanentAddress";
Descriptions = {
    instance of

```

```

Microsoft_CliLocalizedString

```

```

5      {
          CodePage = 401;
          Text = "PermanentAddress
defines the network address hard coded into an adapter. This 'hard
10      coded' address may be changed via firmware upgrade or software
configuration. If so, this field should be updated when the change is
made. PermanentAddress should be left blank if no 'hard coded' address
exists for the network adapter.";

```

```

          });
          Name = "PermanentAddress";
      },

```

```

15      instance of Microsoft_CliProperty
      {

```

```

          Derivation = "PNPDeviceID";
          Descriptions = {
20      instance of

```

```

Microsoft_CliLocalizedString
      {
          CodePage = 401;
          Text = "Indicates the Win32
25      Plug and Play device ID of the logical device. Example: *PNP030b";
          });
          Name = "PNPDeviceID";
      },

```

```

30      instance of Microsoft_CliProperty
      {
          Derivation =

```

```

"PowerManagementCapabilities";
          Descriptions = {
          instance of
35      Microsoft_CliLocalizedString
      {

```

```

          CodePage = 401;
          Text = "Indicates the specific
40      power-related capabilities of the logical device. The array values,
0=\ "Unknown\ ", 1=\ "Not Supported\ " and 2=\ "Disabled\ " are self-
explanatory. The value, 3=\ "Enabled\ " indicates that the power
management features are currently enabled but the exact feature set is
unknown or the information is unavailable. \ "Power Saving Modes Entered
45      Automatically\ " (4) describes that a device can change its power state
based on usage or other criteria. \ "Power State Settable\ " (5) indicates
that the SetPowerState method is supported. \ "Power Cycling Supported\ "
(6) indicates that the SetPowerState method can be invoked with the
PowerState input variable set to 5 (\ "Power Cycle\ "). \ "Timed Power On
50      Supported\ " (7) indicates that the SetPowerState method can be invoked
with the PowerState input variable set to 5 (\ "Power Cycle\ ") and the
Time parameter set to a specific date and time, or interval, for power-
on.";

```

```

          });
          Name = "PowerManagementCapabilities";
      },
55      instance of Microsoft_CliProperty

```





```

        Name = "Speed";
    },
    instance of Microsoft_CliProperty
    {
        Derivation = "Status";
        Descriptions = {
            instance of
Microsoft_CliLocalizedString
            {
10                CodePage = 401;
                Text = "The Status property is
a string indicating the current status of the object. Various
operational and non-operational statuses can be defined. Operational
statuses are \"OK\", \"Degraded\" and \"Pred Fail\". \"Pred Fail\"
15 indicates that an element may be functioning properly but predicting a
failure in the near future. An example is a SMART-enabled hard drive.
Non-operational statuses can also be specified. These are \"Error\",
\"Starting\", \"Stopping\" and \"Service\". The latter, \"Service\",
could apply during mirror-resilvering of a disk, reload of a user
20 permissions list, or other administrative work. Not all such work is on-
line, yet the managed element is neither \"OK\" nor in one of the other
states.\";

                }};
        Name = "Status";
    },
25    instance of Microsoft_CliProperty
    {
        Derivation = "StatusInfo";
        Descriptions = {
            instance of
30    Microsoft_CliLocalizedString
            {
                CodePage = 401;
                Text = "StatusInfo is a string
35 indicating whether the logical device is in an enabled (value = 3),
disabled (value = 4) or some other (1) or unknown (2) state. If this
property does not apply to the logical device, the value, 5 (\"Not
Applicable\"), should be used.\";

                }};
        Name = "StatusInfo";
    },
40    instance of Microsoft_CliProperty
    {
        Derivation = "SystemCreationClassName";
        Name = "SystemCreationClassName";
    },
45    instance of Microsoft_CliProperty
    {
        Derivation = "SystemName";
        Name = "SystemName";
    },
50    instance of Microsoft_CliProperty
    {
        Derivation = "TimeOfLastReset";
        Descriptions = {
55

```

```

instance of
Microsoft_CliLocalizedString
{
    CodePage = 401;
    Text = "The TimeOfLastReset
property indicates when the network adapter was last reset.";
    };
    Name = "TimeOfLastReset";
    };
    };
    FriendlyName = "Win32_NetworkAdapter";
    Target = "Select * from Win32_NetworkAdapter";
    Verbs = {
        instance of Microsoft_CliVerb
        {
            Derivation = "SetPowerState";
            Descriptions = {
                instance of Microsoft_CliLocalizedString
                {
                    CodePage = 401;
                    Text = "SetPowerState defines the desired
power state for a logical device and when a device should be put into
that state. The desired power state is specified by setting the
PowerState parameter to one of the following integer values: 1=\"Full
Power\", 2=\"Power Save - Low Power Mode\", 3=\"Power Save - Standby\",
4=\"Power Save - Other\", 5=\"Power Cycle\" or 6=\"Power Off\". The Time
parameter (for all state changes, except 5, \"Power Cycle\") indicates
when the power state should be set, either as a regular date-time value
or as an interval value (where the interval begins when the method
invocation is received). When the PowerState parameter is equal to 5,
\"Power Cycle\", the Time parameter indicates when the device should
power on again. Power off is immediate. SetPowerState should return 0 if
successful, 1 if the specified PowerState and Time request is not
supported, and some other value if any other error occurred.";
                };
                Name = "SetPowerState";
                Parameters = {
                    instance of Microsoft_CliParam
                    {
                        ParaId = "PowerState";

                        Type = "UINT16";
                        Descriptions;

                    },
                    instance of Microsoft_CliParam
                    {
                        ParaId = "Time";

                        Type = "DATETIME";
                        Descriptions;

                    }
                };
                Usages = {

```

```

        instance of Microsoft_CliLocalizedString
        {
            CodePage = 401;
            Text = "SetPowerState";
        };
    },
    instance of Microsoft_CliVerb
    {
        Derivation = "Reset";
        Descriptions = {
            instance of Microsoft_CliLocalizedString
            {
                CodePage = 401;
                Text = "Requests a reset of the logical
15 device. The return value should be 0 if the request was successfully
                executed, 1 if the request is not supported and some other value if an
                error occurred.";
            };
        Name = "Reset";
        Usages = {
            instance of Microsoft_CliLocalizedString
            {
                CodePage = 401;
                Text = "Reset";
            };
        };
    };
};

```

## APPENDIX B

### EXAMPLE XSL FILE

In an exemplary system environment **200** as shown in **Fig. 2**, the following XSL files are used to format the display for a list of properties returned through a command alias:

#### Example 1: (WmiCmdTableFormat.xsl)

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/" xml:space="preserve">
    Name: Value:
    <xsl:for-each select="CIM//INSTANCE/PROPERTY">
      <xsl:value-of select="@NAME" />
      :
      <xsl:value-of select="VALUE" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

#### Example 2: (WmiCmdValueFormat.xsl)

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/" xml:space="preserve">
    <xsl:for-each select="CIM//INSTANCE/PROPERTY">
      <xsl:value-of select="VALUE" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```



<verb parameter>	::=	<actual parameter>   <actual parameter> , <verb parameter>
<standard verb>	::=	<get verb>   <list verb>   <assoc verb>   <call verb>   <set verb>
<identifier>	::=	<idhead> [<idrest>]
<idhead>	::=	<letter>
<idrest>	::=	<identifier> [<letter>   <digit>]
<get verb>	::=	GET [<property list>] [<get switches>]
<property list>	::=	<property name>   <property name> , <property list>
<list verb>	::=	LIST [<list format> <list switches>]
<assoc verb>	::=	ASSOC [<format specifier>]
<call verb>	::=	CALL <method name> [<actual parameter list>]
<actual parameter list>	::=	<actual parameter>   <actual parameter> , <actual parameter list>
<set verb>	::=	SET <assign list>
<assign list>	::=	<property name> = <property value>   <property name> = <property value> <assign list>
<get switches>	::=	/VALUE   /ALL   /TRANSLATE   /EVERY :<interval>   /FORMAT [:<format specifier>]   /DESCRIPTION [:<code page>]
<interval>	::=	!! numeric value indicating frequency within which values should be returned
<formatspecifier>	::=	:<xsl file name>   :TABLE   :MOF
<list format>	::=	BRIEF   INSTANCE   SYSTEM   STATUS   FULL   <user format>
<list switches>		/TRANSLATE   /EVERY :<interval>   /FORMAT [:<format specifier>]
<help type>	::=	: BRIEF   : FULL

## APPENDIX D

### EXAMPLES OF COMMAND LINE PROCESSING

#### General

##### A.1 Global Switches (Qualifiers) Usage:

5 Global switches (to the right of the "\$ wmic") denote operations that operate at the full context of the command. Therefore, these switchers apply to an entire session established by the command.

**\$ wmic /?**

10 Display command global switches and all registered aliases.

**\$ wmic /locale 40b**

*specify Finnish for localization (impersonation)*

**\$ wmic /NODE**

15 Which node to connect to for getting info

**\$ wmic /NODE /?** //Discover mgmt arenas <based on namespaces>  
defaults to \\root\\cli>

20 Network - manage the network subsystem namespace  
Apps - manage applications namespace  
System - manage operating system namespace  
Devices- manage devices namespace  
Users - manage users namespace  
25 DB - manage DBAs namespace

**\$ wmic /NODE \\root\\cimv2** //Escape to preferred namespace

30 **\$ wmic /NODE \\remoteServerA \\root\\cimv2\\applications**  
set operations against aliases on the specified namespace of **remoteServerA**  
**\\root\\cimv2\\applications**

35 **\$ wmic /NODE system /?** //Discover any available sub scopes

40 Processor - manage the network subsystem <alias>  
Bios - manage BIOS functions  
Disks - manage storage  
LogDrives - manage logical drive partitions  
Process - manage operating processes  
Service - manage system services  
DCOM - manage DCOM Configuration  
Scheduling - manage jobs

**\$ wmic /trace**  
output all debugging info to {stderr}

5

## A.2 Command Line Alias

An example of a printer alias is as follows:

**\$ wmic printer**  
# Where 'printer' is defined as an alias for WIN32\_PRINTER is  
equivalent to the following class escape:

**\$ wmic CLASS WIN32\_PRINTER**

10

15

## A.3 Verb Usage

### Standard Verb Operations

**\$ wmic {alias} {verb} /?**

20

[Display description, switches, and parameters]  
[Display description, verb, and Keyword info for the specified alias. Verbs  
available to aliases (*Only supported standard verbs for an alias will be shown*) include:

<b>GET</b>	Data get operations
<b>SET</b>	Data set operations
<b>CALL</b>	Method, execution operations
<b>LIST</b>	Show data (like netsh, etc.)
<b>ASSOC</b>	Associate operation/data according to specified format.

25

### GET Operations

**\$ wmic {alias} GET /?**

30

Display get switches and all properties for the specified object and its  
descriptions. If the object was an ALIAS - only the properties that are defined  
for that alias are displayed (**whether view object, scopes, containers, etc.**).

These properties map to properties on the referred to WMI OBJECT but can have  
different (user friendly) names. If the object was a WMI CLASS NAME  
(eg. wmic CLASS WIN32\_PRINTER GET /?) then the properties come from the  
class itself.

35

GET SWITCHES include:

<b>/VALUE (default)</b>	Return value (mapped if required)
<b>/DESCRIPTION</b>	Return description
<b>/ALL</b>	Return the data and metadata for attribute
<b>/TRANSLATE</b>	Translate return value via UNIX TR semantics. Useful for exporting to CSVs
<b>/EVERY</b>	Return values every ( X interval) seconds

40



**/FORMAT**

Keyword OR XSL filename to process XML  
Results.

Example of a user-friendly command:

**\$ wmic system\process GET /FORMAT :TABLE \* EXECUTABLEPATH**

```
401  E:\\Program Files\\Internet Explorer\\IEXPLORE.EXE
402  E:\\WINNT\\system32\\psxss.exe
1232 E:\\WINNT\\system32\\msiexec.exe
```

A user-friendly command:

**\$ wmic system\process GET /DESCRIPTION : 401 ExecutablePath**

**PageFaults**

*PageFileUsage PageFileUsage*

HANDLE 401 A string used to identify the process. A process  
ID is a process handle.

ExecutablePath E:\\Program Files\\Internet Explorer\\IEXPLORE.EXE  
The ExecutablePath property indicates the path  
to the executable file of the process

PageFaults 3062 The PageFaults property indicates the number  
of page faults generated by the process.

The user-friendly command:

**\$ wmic system\process WHERE (Caption = "SPOOLSV.EXE") GET Threads , Faults  
/FORMAT : TABLE /EVERY :15**

```
532  10  3062
532  12  3093
532  11  3102
```

...

**# A new line is printed every 15 seconds**

Is equivalent to the following:

**\$ wmic CLASS WIN32\_PROCESS WHERE (Caption = "SPOOLSV.EXE") GET  
ThreadCount, PageFaults /FORMAT :TABLE /EVERY: 15**

```
532  10  3062
532  12  3093
532  11  3102
```

...

**# Note that different property names from the previous example.  
The previous example had # user-friendly names but this refers to  
the WMI class so it uses its property names.**

## SET Operations

The interactive command syntax:

**\$ wmic users\accounts WHERE(domain=Redmond AND disabled=false AND locked=true) SET disabled=true /INTERACTIVE**

disable account Redmond\Travism[y/n]y  
account disabled  
disable account Redmond\UserJoe[y/n]n  
account skipped  
disable account Redmond\j-mier[y/n]y  
account disabled

## LIST Operations

**\$ wmic {alias} LIST /?**

[Display swithes and options. Switches include:

<b>/FULL</b>	Return the full set of properties
<b>/BRIEF</b>	Return a BRIEF set of the properties
<b>/INSTANCE</b>	Return just the instance names
<b>/TRANSLATE</b>	Translate return value via UNIX TR semantics. Useful for exporting to CSVs
<b>/EVERY</b>	Return values every X (specified interval) seconds
<b>/SYSTEM</b>	Show system properties
<b>/FORMAT</b>	Specify an XSL to format data
<b>/STATUS</b>	Show the status of the object
<b>/CONFIG</b>	Return the configuration of the component
<b>&lt;user format&gt;</b>	Show what the user format is.

The user-friendly command syntax:

**\$ wmic system\process LIST**

HANDLE	NAME	PATH
=====	=====	=====
401	IEXPLORE.EXE	E:\\Program Files\\Internet Explorer\\IEXPLORE.EXE
402	psxss.exe	E:\\WINNT\\system32\\psxss.exe
1232	msiexec.exe	E:\\WINNT\\system32\\msiexec.exe
1103	svchost.exe	E:\\WINNT\\System32\\svchost.exe
...		

The user-friendly command syntax, using BRIEF qualifier:

**\$ wmic system\process 1103 LIST brief**

Handle 1103

```

Name svchost.exe
ExecutionPath E:\WINNT\System32\svchost.exe
PageFaults 3062
PageFileUsage 3481600
...

```

The user-friendly command syntax, using FULL switch:

```

$ wmic system\process 401list full
Caption = "notepad.exe";
CreationClassName = "Win32_Process";
CreationDate = "20000414150141.596597-420";
CSCreationClassName = "Win32_ComputerSystem";
CSName = "JSNOVER004";
Description = "notepad.exe";
ExecutablePath = "E:\\WINNT\\System32\\notepad.exe";
Handle = "1172";
HandleCount = 21;
KernelModeTime = "36852992";
MaximumWorkingSetSize = 1413120;
MinimumWorkingSetSize = 204800;
Name = "notepad.exe";
OSCreationClassName = "Win32_OperatingSystem";
OSName = "Microsoft Windows 2000 Professional
[E:\\WINNT\\Device\\Harddisk0\\Partition2";
OtherOperationCount = "9";
OtherTransferCount = "0";
PageFaults = 299;
PageFileUsage = 282624;
ParentProcessId = 288;
PeakPageFileUsage = 290816;
PeakVirtualSize = "14680064";
PeakWorkingSetSize = 1105920;
Priority = 8;
PrivatePageCount = "282624";
ProcessId = 401;
QuotaNonPagedPoolUsage = 1876;
QuotaPagedPoolUsage = 17284;
QuotaPeakNonPagedPoolUsage = 1928;
QuotaPeakPagedPoolUsage = 17988;
ReadOperationCount = "0";
ReadTransferCount = "0";
SessionId = 0;
ThreadCount = 1;
UserModeTime = "4907056";
VirtualSize = "14667776";
WindowsVersion = "5.0.2195";
WorkingSetSize = "1101824";
WriteOperationCount = "0";

```

WriteTransferCount = "0";

**\$ wmic system\process where (name = svchost.exe) list full**

Handle 1103  
Name svchost.exe  
ExecutionPath E:\WINNT\System32\svchost.exe  
PageFaults 3062  
PageFileUsage 3481600  
...

### **Advanced Scenarios Available to the User**

*I want to be able to list printers on a server, view status like out of paper...*

**\$ wmic /NODE:servername devices\printer LIST DetectedErrorState**

Name	Port	DetectedErrorState
====	====	=====
HP LaserJet 4Si	LPT1:	No Error
\\ntprint\By DaveTh	40_hall2 npide99b9	No Paper
\\corp1\ntprinter2	44_4 aod444	Low Toner

+++++

*I want to be able to pause or delete jobs, change properties...*

**\$ wmic printjob Where(Name=\\corp1\ntprinter2 and Size > 1000000) kill /interactive**

146 Microsoft Word - DVD-RAM.doc [y/n]y  
deleted  
147 Microsoft Word - Life of Brian.doc[y/n]n  
148 Microsoft Word - Whistler Plan[y/n]Y  
deleted

+++++

*I also want to restrict certain printer description to 48 chars for DOS clients even though schema allows 256.*

**\$ wmic CLASS WIN32\_printer WHERE (Name=HP LaserJet 4Si) SET DESCRIPTION "this is a test of the length of a description"**

**\$ wmic printer WHERE (Name=HP LaserJet 4Si) SET DESC "this is a test of the length of a description"**

ERROR: Text ("this is a test of the length of a description") too long

**#Note that the first one succeed and it's name was "Description" the second one failed and it's name was DESC. This is because the Alias specified as PROPERTY DESC that mapped to DESCRIPTION but had**

additional semantic/restrictions. In this case it had a short MAX\_LENGTH

+++++//  
need help occasionally because i can't always remember the syntax...

**\$ wmic /ROLE/?**

Network  
Apps  
System  
Devices  
Users  
DB

**\$ wmic system /?**

Processor  
Bios  
Disks  
LogDrives  
Process  
Service  
DCOM  
Scheduling

**\$ wmic system\printer /?**

PRINTER

GET - Get parameters  
SET - Set Parameters  
CYCLE - Cycle Power  
SETPOWERSTATE - Set power state  
.....

**\$ wmic system\printer GET /?**

Property	Type	Operation
====	====	=====
Availability	INT16	Read-Only
AveragePagesPerMinute	INT32	Read-Only
Caption	String	ReadWrite
DriverName	String	Read-Only
PortName	String	Read-Only
...		

**\$ wmic system\printer SET /?**

Property	Type
====	====
Caption	String
...	

**\$ wmic system\printer CALL /?**

Call	Input Param(s)&Type	Status
=====	=====	=====
reset		Implemented
setpowerstate	powerstate(int16) time(date/time)	Implemented

*Verifies in WMI schema qualifier if the method is implemented.*

**\$ wmic system\printer CALL /? : brief**

Call	Input Param(s)&Type	Status
=====	=====	=====
reset		Implemented
setpowerstate	powerstate(int16) time(date/time)	Implemented

**\$ wmic system\printer GET /? : brief**

Property	Type	Operation
====	====	=====
Availability	INT16	Read-Only
AveragePagesPerMinute	INT32	Read-Only
Caption	String	Read\Write
DriverName	String	Read-Only
PortName	String	Read-Only
...		

**\$ wmic system\printer GET \\corp00\ntprint1 Availability /? : full**

Property	Type	Operation
====	====	=====
Availability	INT16	Read-Only

**Description:**

The availability and status of the device. For example, the Availability property indicates that the device is running and has full power (value=3), or is in a warning (4), test (5), degraded (10) or power save state (values 13-15 and 17). Regarding the power saving states, these are defined as follows: Value 13 ("Power Save - Unknown") indicates that the device is

known to be in a power save mode, but its exact status in this mode is unknown; 14 ("Power Save - Low Power Mode") indicates that the device is in a power save state but still functioning, and may exhibit degraded performance; 15 ("Power Save - Standby") describes that the device is not functioning but could be brought to full power 'quickly'; and value 17 ("Power Save - Warning") indicates that the device is in a warning state, though also in a power save mode.

**\$ wmic system/printer CALL !? : full**

<i>Call</i>	<i>Input Param(s)&amp;Type</i>	<i>Status</i>
=====	=====	=====
<i>reset</i>		<i>Implemented</i>

**Description:**

Requests a reset of the logical device. The return value should be 0 if the request was successfully executed, 1 if the request is not supported and some other value if an error occurred.

<i>Call</i>	<i>Input Param(s)&amp;Type</i>	<i>Status</i>
=====	=====	=====
<i>setpowerstate</i>	<i>powerstate(int16)</i> <i>time(date/time)</i>	<i>Implemented</i>

**Description:**

SetPowerState defines the desired power state for a logical device and when a device should be put into that state. The desired power state is specified by setting the PowerState parameter to one of the following integer values: 1="Full Power", 2="Power Save - Low Power Mode", 3="Power Save - Standby", 4="Power Save - Other", 5="Power Cycle" or 6="Power Off". The Time parameter (for all state changes, except 5, "Power Cycle") indicates when the power state should be set, either as a regular date-time value or as an interval value (where the interval begins when the method invocation is received). When the PowerState parameter is equal to 5, "Power Cycle", the Time parameter indicates when the device should power on again. Power off is immediate. SetPowerState should return 0 if successful, 1 if the specified PowerState and Time request is not supported, and some other value if any other error occurred.